Tekstur i titanlegeringer

Kaat, G.A.

Publication date:
2002

Document Version
Publisher's PDF, also known as Version of record

Risø-R-1361(DA)

Tekstur i titanlegeringer

Gregers Alexander Kaat

Forskningscenter Risø, Roskilde
August 2002

Resume
Eksisterende software til behandling af EBSD-data er blevet videreudviklet med henblik på at kunne håndtere hcp-materialer. Det specifikke problem har været tekstur i tynde titanium-plader, som på baggrund af røntgenundersøgelser forventedes at have en svag tekstur.
Det udviklede program har kunne håndtere målte EBSD data og plotte disse som polfigurer. Nye EBSD-målinger er blevet udført på titanium-legeringer leveret af Alfa Laval, og resultaterne viser en kraftig valsetekstur i legeringerne.

ISBN 87-550-3107-2
ISBN 87-550-3108-0 (Internet)
ISSN 0106-2840

Print: Pitney Bowes Management Services Denmark A/S, 2002

Indhold

Forord

Denne rapport er udarbejdet i forbindelse med et to måneders sommerprojekt udført i Afdelingen for Materialeforskning, Forskningscenter Risø. Med udgangspunkt i tidligere teksturundersøgelser af titanlegeringer har projektet sigtet mod udvikling af computersoftware til præsentation af data målt ved hjælp af elektrontilbagespredningsdiffraktion (EBSD). Programmeringsarbejdet har været baseret på N.C.Krieger-Lassen's program PFPLOT, som anvendes til at plotte EBSD-data i såkaldte polfigurer. Programmet skulle udvides til at kunne behandle hcp-materialer og ikke blot fcc/ bcc-materialer. Dette er relevant, fordi titanium er et hcp-materiale. Projektet er gennemført under vejledning af Søren Schmidt og Jesper Vejlø Carstensen.

1 Introduktion

En krystalstruktur består af enhedsceller, f.eks. kubiske (fcc) eller hexagonale (hcp). Over små volumener af materialet vil cellerne have samme orientering; disse volumener kaldes korn. Kornene kan være orienteret mere eller mindre tilfældigt og materialet siges da at have en svag hhv. stærk tekstur.

En enhedscelle karakteriseres ved sin symmetrigruppe, dvs. den mængde af drejninger og spejlinger, som afbilleder cellen identisk. Givet en vektor kaldes de vektorer, som opnås ved at bruge funktionerne i symmetrigruppen på den, for symmetriækvivalente. Dette definerer de symmetriækvivalente retninger og planer i krystallen. For at udnytte krystallens symmetri i behandlingen af problemer vedrørende refleksioner mod krystalplanerne, vælges som koordinatsystem enhedscellens akser. Man opererer med såkaldte Miller-indices for krystalretninger og –planer, betegnet hhv. [h k l] og (h k l). Sæt af ækvivalente retninger hhv. planer betegnes <h k l> hhv. {h k l}, med en repræsentant for ækvivalensklassen givet ved [h k l] hhv. (h k l). Mere detaljeret information om krystallografi kan f.eks. findes i [1]

I hcp-struktur tilføjes en overflødig 4. akse i den hexagonale plan for at udnytte symmetri til at lette beregningen af ækvivalente retninger og planer; koordinaterne her er [h k i l] hhv. (h k i l), og man pålægger betingelsen h + k + i = 0, idet h, k i er koordinater til tre akser i xy-planen (dvs. een overflødig) med 120 grader imellem sig; 4.-koordinaten l er højden af cellen, normal på de tre andre akser. Akserne i xy-planen er af samme længde og betegnes a1, a2, a3, højdeaksen betegnes c. For a = a1, a2 eller a3 er ratioen |c|/|a| af betydning for tekstur i materialer, som er blevet rullet – se [2].

På scanningelektronmikroskopet (SEM) foretages automatiske EBSD (electron back scattering diffraction) målinger vha. programmet CROMATIC. Resultatet af en sådan måling er en såkaldt CRO-fil, som for punkter med en given afstand i x- og y-retningerne på materialets overflade indeholder flg. informationer: 1) eulervinkler som beskriver rotationen af de celler, som elektronerne spredes fra, og 2) antal refleksioner, som idenificeres. Rotationen beregnes af CROMATIC ved at regne på den spredte stråling, som detekteres. Antal identificerede refleksioner (ud af et maksimalt forventet antal) svarer til sandsynligheden for, at identifikationen er rigtig [3].

Programmet PFPLOT afbilleder resultatet af en EBSD-måling som en polfigur, forudsat materialet er fcc eller bcc [4]. Det nye program, skrevet i MATLAB, kan lave polfigurer for både bcc/ fcc og hcp-materialer. En pol er defineret som en fladenormal gennem (0,0,0). En polfigur er et plot af krystalplannormalerne og fremstilles således:

For hvert målepunkt checkes det, at antal identificerede refleksioner er over et vist tal; hvis ikke, så smides målepunktet væk. For hvert ikke-kasseret målepunkt haves en rotationsmatrix, beregnet ud fra eulervinklerne. Udfra viden om symmetrigruppen beregner programmet fladenormaler til ækvivalente krystalplaner, idet brugeren blot skal angive indices til et enkelt plan. For hver rotationsmatrix (dvs. hvert målepunkt) roteres mængden af fladenormaler i rummet. For hver fladenormal fås således en orientering i rummet, svarende til en vektor fra (x,y,z) = (0,0,0), hvis forlængelse skærer enhedskuglen i en punkt. Hvis dette punkt P ligger under xy-planen (z negativ), erstattes P med –P. Punktet projiceres ned i xy-planen vha. den stereografiske projektion, dvs. langs linien, som forbinder P med (0,0,-1). Således afbilledes fladenormalerne på

enhedscirklen, her kaldet polplanen, på en sådan måde, at næsten ens orienteringer ligger nær hinanden. Bemærk at modsatte punkter på periferien identificeres med hinanden.

Hvis målingen omfatter et tilpas stort antal korn, vil man af polplottet kunne aflæse materialets kornorienteringer; hvis disse klumper sig sammen omkring få punkter på polfiguren, har materialet stærk tekstur, og hvis de er tilfældigt fordelt, har materialet svag eller ingen tekstur svarende til tilfældig kornorientering.

Krystalplannormalerne afbilledes på tre måder: Som punkter i polplanen, eller som to slags densitetsplots (i polplanen); sidstnævnte er konturplot eller farveplot. Densitetsplottene beregnes udfra punkterne på polsfæren, idet en tæthedsfunktion beregnes her og projiceres ned i polplanen. Tæthedsfunktionen er afledt af Watsonfordelingen. Hvor meget, man ønsker at 'glatte ud' i overgangen fra punktplot til densitetsplot, afhænger strengt taget af variansen af punkterne, men i programmet er udglatningsparameteren gjort brugerdefineret.

Som kordinatsystem anvendes RD, TD, ND, idet RD = rolling direction = den retning, materialet er blevet rullet, TD = transverse direction = den på RD vinkelrette retning i materialets overfladeplan, ND = retningen normal på materialets overflade. Den hexagonale enhedscelles udgangsposition (rotationsmatrix = identitetsmatricen) er defineret så c-aksen ligger langs ND.

Plotning af polfigurer er blot en måde at repræsentere de målte EBSD data på. En anden måde er at plotte data som såkaldte orienteringsfordelingsfunktioner, eller ODF (Orientation Distribution Functions). Dette er et to-dimensionelt plot af intensiteterne målt i Eulerrummet givet ved Eulervinkler φ_1, Φ og φ_2. Til dette bruges programmet ODFplot [5], der plotter φ_1 som funktion af Φ i 5° snit af φ_2 (dvs. φ_2=0, 5°, 10°,….90°). Intensiteterne normeres i forhold til tilfældig tekstur (som tildeles intensiteten 1). Dvs. at maksimale intensiteter der afviger meget fra 1 er et udtryk for kraftig tekstur.

En tredje måde at plotte data på er som såkaldte orienteringskort. Dette sker v.h.a. programmet CROplot [6], som plotter den krystallografiske orientering for hvert målepunkt og sammenholder denne måling med nabomålingen. På den måde fås et billede af, hvordan orienteringen varierer henover det scannede område på prøven, og det bliver muligt af danne et billede af materialets kornstruktur, hvor de enkelte korn tildeles forskellige farvekoder.

2 Eksperimentelle detaljer

2.1 Udvikling af software

Denne del var den mest tidskrævende. Opbygningen af det nyudviklede Matlab-program, kaldet POLEFIG.m, er som følger:

1) Indledende forklarende tekst, hvor brugeren selv skal definere sine variable i selve koden. (Dette er smart i Matlab 6, hvis man skal køre programmet flere gange.)
2) Indlæsning af data fra cro-fil. Udsmidning af dårlige punkter, svarende til at CROMATIC kunne identificere under halvdelen af retningerne i den pågældende gruppe.
3) Beregning af ækvivalente plannormaler (i udgangspositionen, dvs. ingen rotation), afhængig af, om vi er i kubisk eller hexagonal symmetri.
4) Beregning af polerne for hvert målepunkt (dvs. for givne rotationer af enhedscellen).
5) Evt. beregning af densityplots.
6) Tegning af figurer.

I øvrigt henvises til programkoden og kommenteringen i denne (se Appendix).

2.2 Materiale

Titaniumlegeringerne blev leveret af firmaet Alfa Laval i Lund, Sverige. Der blev leveret to tynde plader med påtegnelserne CN1924:1 og CN1924:2, og disse betegnes i det følgende som hhv. Materiale 1 og Materiale 2. Der medfulgte ingen informationer om hvilken termomekanisk behandling disse materialer havde været udsat for, er heller var der oplyst noget om forventningerne m.h.t. tekstur i materialerne.

2.3 Forberedelse af prøver til EBSD

Titaniumlegeringerne slibes ved str. 1000, diamandslibes ved 3 μm og elektropoleres vha. elektrolyt A3 i 25-28 sek. ved 1,0-1,1 A og flowrate 3,5.

Prøverne fastgøres til EBSD-holderen vha. elektrisk ledende cement.

2.4 Målingerne

EBSD-scan blev foretaget vha. et JSM840 scanningelektronmikroskop udstyret med NORDIF EBSD-detektor og HAMAMATSU ARGUS billedbehandlings-udstyr. Programmet CROMATIC blev anvendt til automatisk måling og opsamling af EBSD data.

For hvert materiale blev der forberedt prøver og foretaget en række EBSD 2D-scan med afstand 0.01 eller 0.05 mm i x- og y-retningerne, hvert scan omfattende nogle tusind målepunkter.

Af 0.01 mm målingerne kunne formen og størrelsen af kornene visualiseres vha. programmet CROPLOT. Idet en typisk korndiameter lod til at være lidt under 100 μm, benyttedes afstand 50 μm mellem målepunkterne i de videre målinger (idet man til teksturbestemmelse ideelt ønsker 1 målepunkt pr. korn).

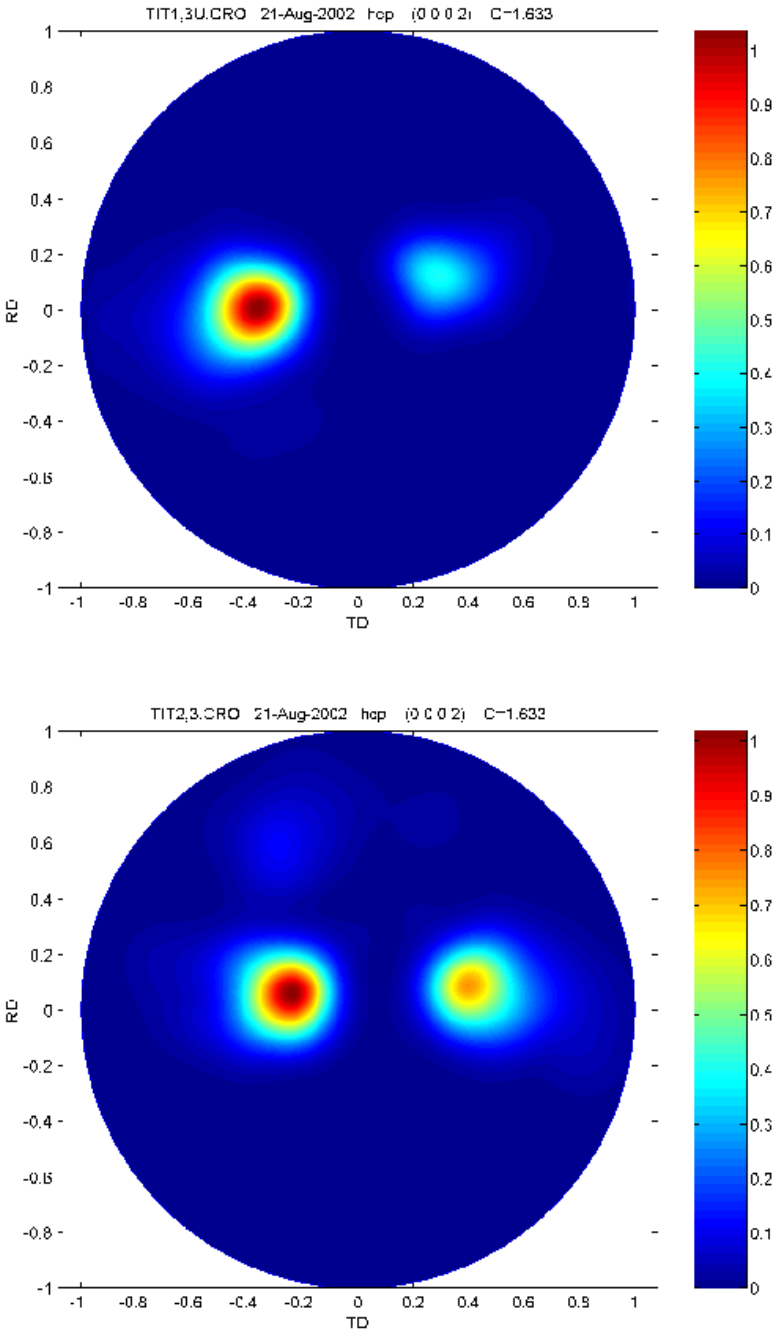
2.5 Databehandling

Hvert EBSD-datasæt blev plottet som farveplots vha. den udviklede software for to sæt af krystalplannormaler, nemlig normalerne til de (0002) ækvivalente planer og til de $(\bar{1}100)$ ækvivalente planer. Desuden anvendtes programmet ODFPlot til at generere et ODF plot for hvert af materialerne.

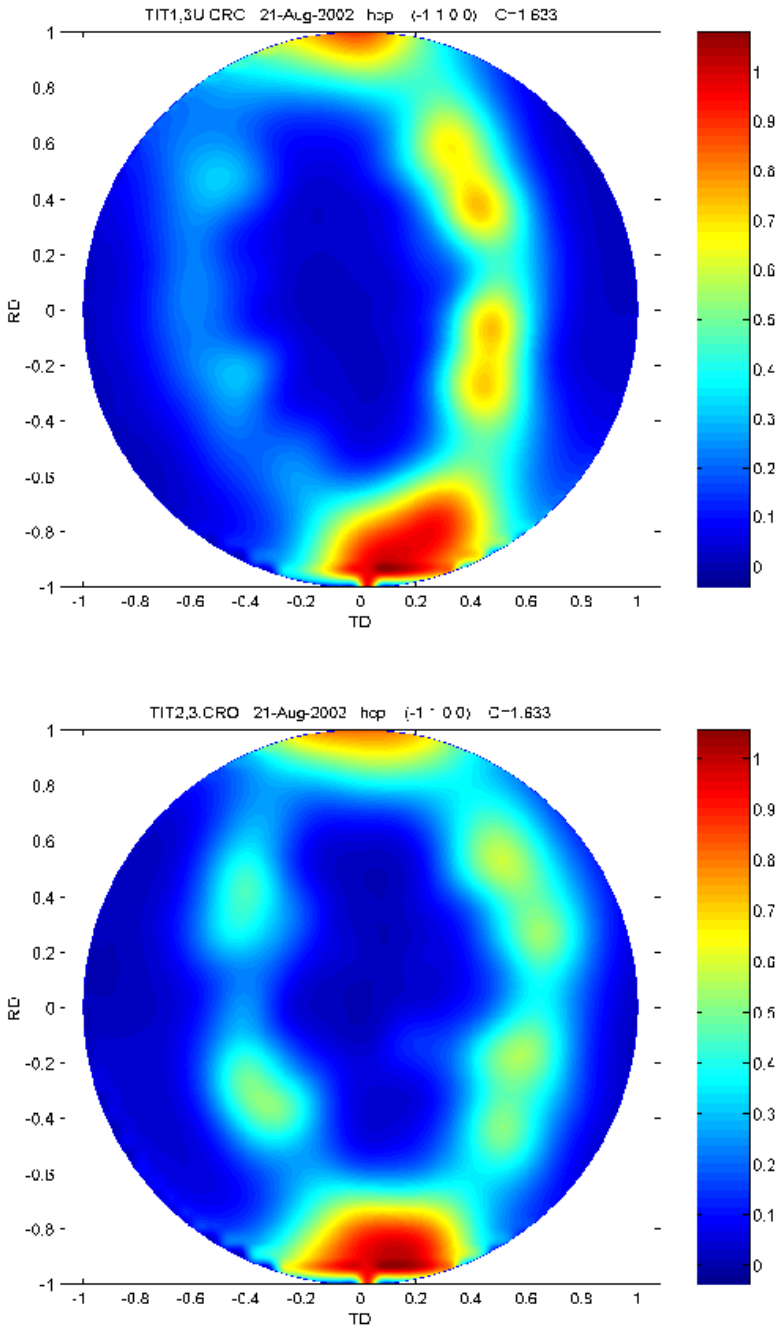
3 Resultater

Nedenfor ses polfigurene plottet med det nyudviklede program. Farveskalaerne på polfigurerne er relative til den maksimale intensitet gående fra 0 til 1.

Polfigurer



Figur 3.1 {0002} polfigurer for Materiale 1 (øverst) og Materiale 2 (nederst).

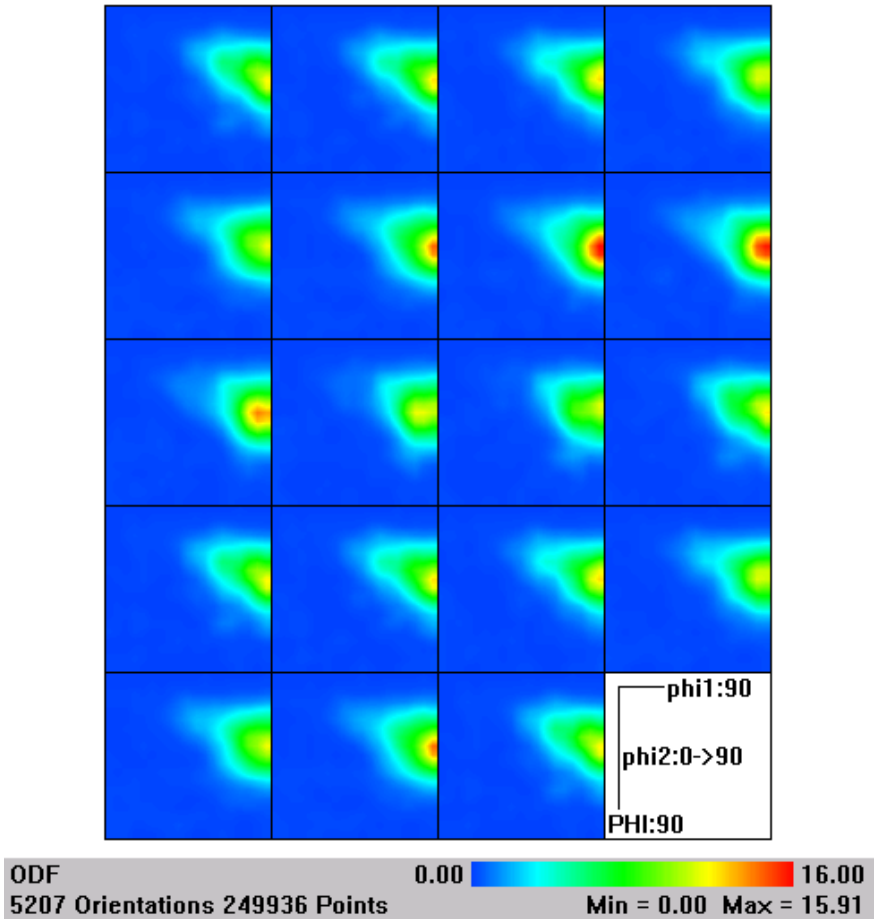


Figur 3.2 $\{\bar{1}100\}$ *polfigurer for Materiale 1 (øverst) og Materiale 2 (nederst).*

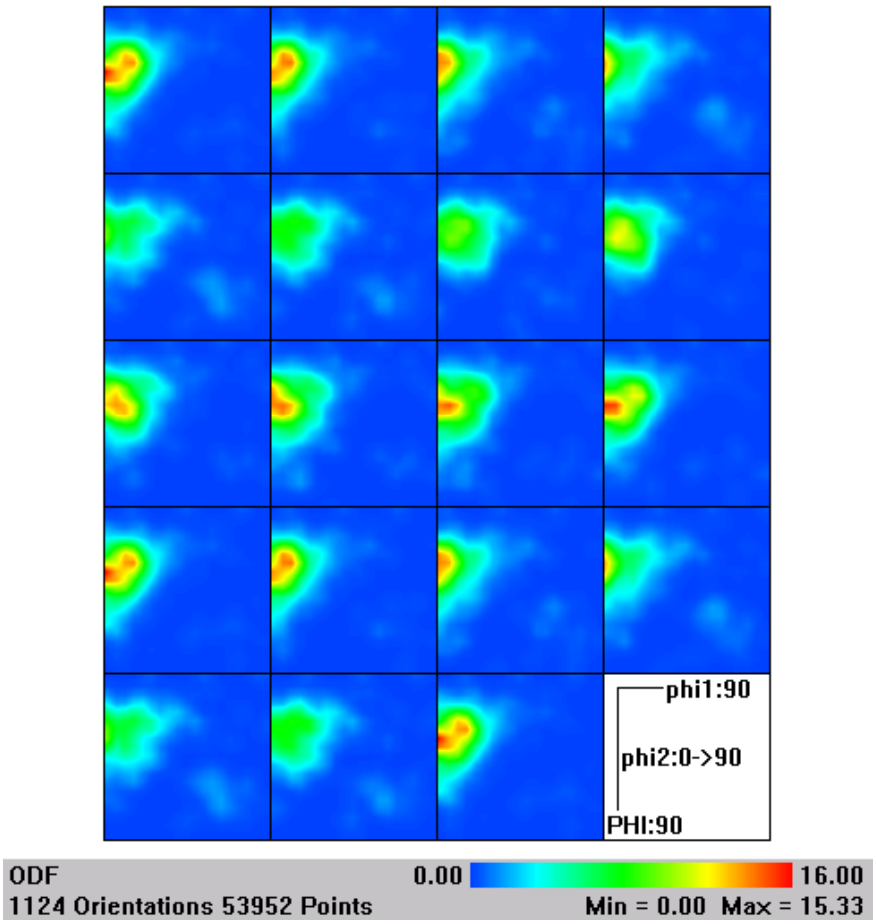
For $\{0002\}$ plannormalerne ses en tydelig tekstur omkring to poler imellem ND og TD. For $\{\bar{1}100\}$ plannormalerne ses ligeledes en symmetri omkring akserne med seks dominerende poler.

ODF plots

Styrken af den maksimale intensitet i forhold til intensiteten ved vilkårlig tekstur fremgår af ODF-plottene nedenfor. Plottene viser en kraftig intensitet omkring (φ_1, Φ) = (90°, 45°) for samtlige φ_2-snit (for materiale 2 er intensiteterne ved φ_1 = 0°, men det skyldes, at prøven fejlagtigt var roteret 90° ved målingen). Dette indikerer en kraftig fibertekstur i materialerne.



Figur 3.3 ODF plot for Materiale 1.



Figur 3.4 ODF plot for Materiale 2.

4 Diskussion

De to plader af titaniumlegeringer lader til at være ret ens mht. tekstur. Intensiteten er maksimal ved hhv. 15,91 og 15,33 gange intensiteten for den vilkårlige tekstur, så teksturen er meget kraftig. Ved sammenligning med andre måleserier ses den maksimale intensitet at variere med ca +/- 2 fra måleserie til måleserie, så der lader ikke til at være forskel på de to materialer mht. styrken af teksturen.

Jvf. [2] gælder det for valsede hcp-materialer at en fibertekstur (givet ved een central pol på $\{0002\}$ polfiguren) kan splitte op i to poler pga. valsningen efter forholdet |c|/|a|: Hvis |c|/|a| < 1,633 (som for titanium) forventes det, at en pol langs ND roteres væk fra ND og mod TD, således at der (ved symmetri) ses to pletter på TD-aksen på polfiguren. Dette ses netop på ovenstående $\{0002\}$ polfigurer. (Hvis |c|/|a| > 1,633 ville man tilsvarende forvente en opsplitning mod RD.)

Iflg. [2] gælder der desuden, at polerne i TD-RD-planen i en hcp fibertekstur ligger som hjørnerne i en hexagon langs randen af $\{\bar{1}100\}$-polfiguren, men at disse poler i det valsede materiale (som ikke har fibertekstur) strækkes ud på polfiguren langs RD når |c|/|a| < 1,633. Dette ses på $\{\bar{1}100\}$-polfigurerne.

5 Konklusioner

Polfigurene som er blevet plottet med det nyudviklede program POLEFIG.m er i overensstemmelse med hvad man kan forvente af et valset hcp materiale. Der observeres en opsplitning af $\{0002\}$ polfigurene langs TD-aksen, og en hexagonal polstruktur observeres på $\{\bar{1}100\}$ polfigurene. Dette viser, at det nyudviklede program POLEFIG.m er i stand til at plotte eksperimentelle EBSD-data for hcp-materialer.

Resultaterne giver udtryk for en kraftig og tilsyneladende ens fibertekstur i de to materialer, hvilket antyder at materialerne har været udsat for valsning.

6 Referencer

[1] Krystallografi….

[2] Rollett & Wright: ”Texture and Anisotropy” side 203 ff.

[3] N.C. Krieger Lassen (2001) *CROMATIC v. 2.10*. Users manual, Risø National Laboratory, Roskilde, Denmark, 13 p.

[4] N.C. Krieger Lassen (2000). *PFPlot v. 1.70*. Users manual, Risø National Laboratory, Roskilde, Denmark, 8 p.

[5] N.C. Krieger Lassen (2000). *ODFPlot v. 1.25*. Users manual, Risø National Laboratory, Roskilde, Denmark, 6 p.

[6] N.C. Krieger Lassen (2000). *CROPlot v. 1.55*. Users manual, Risø National Laboratory, Roskilde, Denmark, 18 p.

Appendix - Kode til POLEFIG.m

Det følgende program er udviklet v.h.a. MATLAB 6.

```
clear PATH FILENAME EXTENSION LATTICETYPE h0 k0 l0 h1 k1 i1 l1 C;
clear POINT_PLOT DENSITY_CONTOUR_PLOT DENSITY_COLOR_PLOT CALC;

%WRITTEN BY GREGERS ALEXANDER KAAT.

%THIS MATLAB PROGRAM CALCULATES A POLE FIGURE FROM THE DATA IN A CRO-FILE
%AND CALCULATES VARIOUS PLOTS. THE PROGRAM CAN TREAT CUBIC (BCC/ FCC) AND
%HEXAGONAL (HCP) STRUCTURES.
%IT'S UP TO THE USER TO SPECIFY THE EQUIVALENCE CLASSES OF UNIT CELL PLANE
%NORMALS TO BE USED.
%THE PROGRAM CALCULATES THE EQUIVALENTS IN THE GIVEN GROUP AS SPECIFIED
%BELOW.
%THE RESULTING POLE FIGURES ARE A REPRESENTATION OF THE TEXTURE OF THE
%GIVEN MATERIAL.

%======= SOME INPUT IS REQUIRED, SO FIRST FILL OUT THE FOLLOWING: =========

%LOCATION AND NAME OF YOUR DATA FILE:
PATH = 'C:\titanium\cro\';
FILENAME = 'TIT2,3';
EXTENSION = '.CRO';

%THE PROGRAM PRESUMES THAT THE 2D-SCAN WAS MADE WITH x = TD, y = RD
%(RD = ROLLING DIRECTION, TD = TRANSVERSE DIRECTION).
%IF IT WAS NOT, THEN SPECIFY THE ANGLE OF ROTATION OF THE xy-PLANE
%OF THE SCAN RELATIVE TO THIS (ANGLE IN DEGREES).
%IE., IF YOU HAVE x = RD, y = TD, THEN SET THETA = 90.
%IF YOU HAVE x = TD, y = RD, SET THETA = 0.
THETA = 90;

%SET LATTICETYPE = 1 FOR CUBIC, 2 FOR HEXAGONAL STRUCTURES:
LATTICETYPE = 2;

%IF CUBIC: DEFINE THE SYMMETRY GROUPS {h k l} OF PLANES, THE NORMALS OF
%WHICH YOU WANT TO PLOT, BY ONE MEMBER OF EACH GROUP. {h k l} CONTAINS
%2, 4, OR 8 DIFFERENT PLANES, GIVEN BY (h k l), (-h k l), (h -k l), ...
%(-h -k -l). IF YOU WANT TO USE THE PLANES {1 2 3} WITH THE PLANES {4 5 6}
%AND {7 8 9}, THEN LET h0 = [1 4 7], k0 = [2 5 8], l0 = [3 6 9].
%YOU PROBABLY WANT TO TRY {1 1 1}: THEN LET h0 = [1], k0 = [1], l0 = [1].
h0 = [1];
k0 = [1];
l0 = [1];

%IF HEXAGONAL: DEFINE THE SYMMETRY GROUPS {h k i l} OF PLANES, THE NORMALS
%OF WHICH YOU WANT TO PLOT, BY ONE MEMBER OF EACH GROUP. {h k i l} CONTAINS
%AS MUCH AS (6 + 6) * 2 = 24 DIFFERENT PLANES, GIVEN BY PERMUTATIONS OF
%h k i AND OF -h -k -i, WITH THE FOURTH COORDINATE +/- l.
%FOR HIGHER INDICES, THERE MAY BE ADDITIONAL SYMMETRY, AND YOU MUST RUN
%WITH MORE THAN ONE SET OF PLANES, EQUIVALENT IN 'BARE' HEXAGONAL SYMMETRY.
%IF YOU WANT TO USE THE PLANES {1 2 3 4} WITH THE PLANES {5 6 7 8} AND
%{9 10 11 12}, THEN LET h0 = [1 5 9], k0 = [2 6 10], i0 = [3 7 11],
%l0 = [4 8 12]. IF YOU JUST WANT ONE SET OF PLANES {1 2 3 4}, LET
%h0 = [1], k0 = [2], i0 = [3], l0 = [4].
%YOU MAY WANT TO TRY {0 0 0 2} AND {0 1 -1 0} (SEPARATELY).
%REMEMBER THAT h1 + k1 + i1 = 0!
h1 = [0];
k1 = [1];
i1 = [-1];
l1 = [0];

%IF HEXAGONAL: ALSO DEFINE THE RATIO C = |c| / |a| OF THE PRIMITIVE CELL,
%WHERE a IS AN AXIS IN THE HEXAGON AND c IS THE HEIGHT AXIS. A STANDARD
%VALUE IS C = sqrt(8/3) ~ 1.633:
C = sqrt(8/3);

%WHAT PLOTS DO YOU WANT? THE DENSITY PLOTS TAKE LONGER TO CALCULATE THAN
```

```
%THE POINT PLOT. 1 FOR THE THE ONES YOU WANT, 0 FOR THE ONES YOU DON'T:
POINT_PLOT = 0;
DENSITY_CONTOUR_PLOT = 0;
DENSITY_COLOR_PLOT = 1;

%FOR DENSITY PLOTS SPECIFY THE DEGREE OF 'SMOOTHING OUT' -
%MUST BE LARGER THAN 0 AND LESS THAN 2.3:
SMOOTH = 1;

%YOU CAN SKIP THE INITIAL (POLE PLOT) CALCULATIONS IF YOU HAVE ALREADY RUN
%THE PROGRAM AND HAVE THESE DATA IN MEMORY. IF YOU WANT TO DO THAT SET
%CALC = 0. IF YOU FURTHERMORE WANT TO SKIP THE DENSITY PLOT CALCULATIONS
%(BECAUSE YOU WANT TO DO A DENSITY PLOT FROM CALCULATIONS PERFORMED DURING
%A PREVIOS RUN) SET CALC = -1. OTHERWISE JUST SET CALC = 1:
CALC = 1;

%==========================================================================

if LATTICETYPE ~= 1 & LATTICETYPE ~= 2
  disp('Define LATTICETYPE in the start of the code!')
  break
end



%READ DATA FROM A CRO-FILE INTO MEMORY:
if CALC ~= 0 & CALC ~= -1
  clear A h k i l SYMM normalV g v w pole density;
  fid = fopen([PATH FILENAME EXTENSION],'r');
  A = fscanf(fid, '%f %f %f %*f %*f %*f %*f %d %d %*f %*f %*f %*s %*s %*s %*s %*s %*s %*s %*s %*s
%*s %*s', [5 inf]);
  status = fclose(fid);
  disp(['Opening ', PATH FILENAME EXTENSION,'.']);
end



%POLE FIGURE CALCULATIONS FOR CUBIC CELLS:
%THE SET {h k l} CONTAINS THE EQUIVALENT PLANES, WHICH IN CUBIC SYMMETRY
%ARE GIVEN BY CHANGES OF ANY NUMBER OF SIGNS OF h, k, l (AT MOST 8
%DIFFERENT ONES). THE PLANE NORMALS ARE THE SET OF DIRECTIONS
%< 1/h 1/k 1/l > WITH THE EQUIVALENTS GIVEN BY CHANGES OF SIGNS. ONLY HALF
%OF THOSE ARE RELEVANT, AS [u v w] AND [-u -v -w] CORRESPOND TO THE SAME
%POINT ON THE POLE FIGURE.
if LATTICETYPE == 1 & CALC ~= 0 & CALC ~= -1
hm = []; %DEFINE hm AS AN EMPTY MATRIX
sh0 = size(h0);
nr_eq_cls = sh0(2); %NUMBER OF CLASSES OF EQUIVALENT PLANES
for m = 1:nr_eq_cls
  h = h0(m);
  k = k0(m);
  l = l0(m);
  %AS 0 = -0 THE NUMBER OF EQUIVALENT PLANE NORMALS DEPENDS ON THE NUMBER
  %OF 0'S. +/- [u v w] IS THE SAME POINT ON THE POLE PLOT, SO WE USE
  %ONLY THIRD COORDINATE POSITIVE.
    if h ~= 0 & k ~= 0
      SYMM = [h -h h -h ; k k -k -k ; l l l l];
    end
    if h == 0 & k == 0
      SYMM = [0 ; 0 ; l];
    end
    if h == 0 & k ~= 0
      SYMM = [0 0 ; k -k ; l l];
    end
    if h ~= 0 & k == 0
      SYMM = [h -h ; 0 0 ; l l];
    end
  sSYMM = size(SYMM);
  sSYMM = sSYMM(2); %NUMBER OF EQUIVALENT PLANES IN CURRENT EQ. CLASS
  shm = size(hm);
  shm = shm(2); %KEEPS TRACK OF NUMBER OF DIRECTIONS IN [hm;km;lm] MATRIX
  for n=1:sSYMM %ADDS SYMM TO [hm;km;lm].
```

```
    %THIS WAY WE CREATE A MATRIX WITH PLANE NORMALS AS COLUMNS. THE PLANE
    %NORMAL COORDINATES ARE GIVEN BY RECIPROCALS OF PLANE COORDINATES:
    hm(n+shm) = 1/ SYMM(1,n);
    km(n+shm) = 1/ SYMM(2,n);
    lm(n+shm) = 1/ SYMM(3,n);
    normhmkmlm = norm([hm(n+shm) km(n+shm) lm(n+shm)]);
    hm(n+shm) = hm(n+shm)/normhmkmlm;
    km(n+shm) = km(n+shm)/normhmkmlm;
    lm(n+shm) = lm(n+shm)/normhmkmlm;
  end
end
%h(n), k(n), l(n) ARE REDEFINED TO CONTAIN THE COORDINATES OF THE RELEVANT
%NORMAL VECTORS.
h = hm;
k = km;
l = lm;
clear hm km lm;

disp('Calculating the pole plot for these plane normal directions');
disp('in the orthogonal xyz coordinate system:');
disp([h;k;l]);

length = 1; %LATER NORMALISATION MAKES LENGTH OF CUBIC CELL IRRELLEVANT
%B IS THE MATRIX OF ORTHOGONALISATION; TRIVIAL IN THIS SYMMETRY:
B = (2*pi/length) * [ 1 0 0 ; 0 1 0 ; 0 0 1 ];

end



%POLE FIGURE CALCULATIONS FOR HEXAGONAL CELLS:
%THE HEXAGONAL UNIT CELL HAS 24 SYMMETRIES, BUT ONLY 12 OF THESE ARE
%RELEVANT, AS WE HAVE FOR A DIRECTION v THAT +/- v CORRESPOND TO THE SAME
%POINT ON THE POLE FIGURE. THE REMAINING SYMMETRIES ARE GIVEN BY THE
%PERMUTATIONS OF h, k, i AND THE PERMUTATIONS OF -h, -k, -i.
if LATTICETYPE == 2 & CALC ~= 0 & CALC ~= -1
nr_eq_cls = size(h1);
nr_eq_cls = nr_eq_cls(2); %NUMBER OF CLASSES OF EQUIVALENT PLANES
H=[];
K=[];
I=[];
L=[];
for m=1:nr_eq_cls
  l1(m) = abs(l1(m));
  if h1(m) + k1(m) + i1(m) ~= 0
    disp('Must have h + k + i = 0!')
    break
  end
  %THE EQUIVALENT PLANES ARE [H(n) K(n) I(n) L(n)], n = 1:12:
  if (h1(m) == k1(m) & k1(m) == i1(m)) %ONLY POSSIBLE FOR:
    H = [H 0];
    K = [K 0];
    I = [I 0];
    L = [L l1(m)];
  end
  %IF TWO INDICES AMONG h1(m), k1(m), i1(m) ARE EQUAL WE REARRANGE
  %TO GET h1(m) = k1(m):
  if (h1(m) == i1(m) & k1(m) ~= h1(m)) | (h1(m) == -i1(m) & k1(m) ~= h1(m) & k1(m) ~= i1(m))
%EXCHANGE k1(m), i1(m)
    temp = i1(m);
    i1(m) = k1(m);
    k1(m) = temp;
  end
  if (k1(m) == i1(m) & h1(m) ~= i1(m)) | (k1(m) == -i1(m) & h1(m) ~= k1(m) & h1(m) ~= i1(m))
%EXCHANGE h1(m), i1(m)
    temp = i1(m);
    i1(m) = h1(m);
    h1(m) = temp;
  end

  %WHEN l1(m) ~= 0 THERE IS AT MOST 12 DIFFERENT EQUIVALENT DIRECTIONS:

  if (h1(m) == k1(m) & k1(m) ~=i1(m) & l1(m)~=0)
```

```
    H = [H h1(m) h1(m) i1(m) -h1(m) -h1(m) -i1(m) ];
    K = [K h1(m) i1(m) h1(m) -h1(m) -i1(m) -h1(m) ];
    I = [I i1(m) h1(m) h1(m) -i1(m) -h1(m) -h1(m) ];
    L = [L l1(m) l1(m) l1(m)  l1(m)  l1(m)  l1(m) ];
  end
  if (h1(m) == -k1(m) & k1(m) ~= i1(m) & k1(m) ~= -i1(m) & l1(m) ~=0)
    %THIS MEANS i1(m) = 0.
    H = [H h1(m)  -h1(m) h1(m)  -h1(m) i1(m)  i1(m)  ];
    K = [K -h1(m) h1(m)  i1(m)  i1(m)  -h1(m) h1(m)  ];
    I = [I i1(m)  i1(m)  -h1(m) h1(m)  h1(m)  -h1(m) ];
    L = [L l1(m)  l1(m)  l1(m)  l1(m)  l1(m)  l1(m)  ];
  end
  if h1(m) ~=k1(m) & h1(m) ~= -k1(m) & k1(m) ~=i1(m) & k1(m) ~= -i1(m) & h1(m) ~=i1(m) & h1(m) ~= -
i1(m) & l1(m) ~=0
    H = [H h1(m) h1(m) i1(m) k1(m) i1(m) k1(m) -h1(m) -h1(m) -i1(m) -k1(m) -i1(m) -k1(m) ];
    K = [K k1(m) i1(m) h1(m) h1(m) k1(m) i1(m) -k1(m) -i1(m) -h1(m) -h1(m) -k1(m) -i1(m) ];
    I = [I i1(m) k1(m) k1(m) i1(m) h1(m) h1(m) -i1(m) -k1(m) -k1(m) -i1(m) -h1(m) -h1(m) ];
    L = [L l1(m) l1(m) l1(m) l1(m) l1(m) l1(m)  l1(m)  l1(m)  l1(m)  l1(m)  l1(m)  l1(m) ];
  end

  %WHEN l0(m) = 0 THERE IS AT MOST 6 DIFFERENT EQUIVALENT DIRECTIONS:

  if (h1(m) == k1(m) & k1(m) ~=i1(m) & l1(m) ==0)
    H = [H h1(m) h1(m) i1(m)];
    K = [K h1(m) i1(m) h1(m)];
    I = [I i1(m) h1(m) h1(m)];
    L = [L l1(m) l1(m) l1(m)];
  end
  if (h1(m) == -k1(m) & k1(m) ~= i1(m) & k1(m) ~= -i1(m) & l1(m) ==0)
    %THIS MEANS i1(m)=0
    H = [H h1(m)  h1(m)  0      ];
    K = [K -h1(m) 0      -h1(m) ];
    I = [I 0      -h1(m) h1(m)  ];
    L = [L l1(m)  l1(m)  l1(m)  ];
  end
  if h1(m) ~=k1(m) & h1(m) ~= -k1(m) & k1(m) ~=i1(m) & k1(m) ~= -i1(m) & h1(m) ~=i1(m) & h1(m) ~= -
i1(m) & l1(m) ==0
    H = [H h1(m) h1(m) i1(m) k1(m) i1(m) k1(m)];
    K = [K k1(m) i1(m) h1(m) h1(m) k1(m) i1(m)];
    I = [I i1(m) k1(m) k1(m) i1(m) h1(m) h1(m)];
    L = [L l1(m) l1(m) l1(m) l1(m) l1(m) l1(m)];
  end
end

disp('Calculating the pole plot for normals to these planes');
disp('in the (hkil) hexagonal coordinate system:');
disp([H; K; I; L]);

sH = size(H);
sH = sH(1,2);
sSYMM = sH;

for n=1:sH
%GIVEN (h k i l) AND THAT h, k, i NOT ALL ZERO: THEN AT LEAST TWO OF THEM
%ARE NOT 0. THE NORMAL VECTOR IN CARTESIAN COORDINATES IS THE CROSS PRODUCT
%OF TWO VECTORS IN THE PLANE, IF THEY ARE NOT PARALLEL TO EACH OTHER. THESE
%ARE CALLED V1, V2.

%THE HEXAGONAL RECIPROCAL LATTICE (IN ORTHONORMAL BASIS):
a1 = [1 0 0];
a2 = [-cos(pi/3) cos(pi/6) 0];
c =  [0 0 1/C];
%B-MATRIX ORTHOGONALISES THE LATTICE VECTOR BASIS a1, a2, c:
B = inv([a1; a2; c]');

if H(n) ==0 & K(n) ==0
   normalV(:,n) = [0;0;1];
end

if H(n)~=0 & K(n) ~=0
  V1 = ([1/H(n); 0 ;0 ] - [0; 1/K(n); 0]);
  if L(n)==0
     V2 = [0; 0 ;1];
  end
```

```
 if L(n) ~=0
    V2 = ([1/H(n); 0;0 ] - [0 ;0; 1/L(n)]);
 end
 normalV(:,n) = cross(inv(B)*V1,inv(B)*V2);
end

if H(n) ~=0 & K(n) == 0
   V1 = [0; 1; 0];
   if L(n) == 0
      V2 = [0; 0;1];
   end
   if L(n) ~= 0
       V2 = ([1/H(n); 0;0 ] - [0 ;0; 1/L(n)]);
   end
   normalV(:,n) = cross(inv(B)*V1,inv(B)*V2);
end

if H(n) ==0 & K(n) ~= 0
   V1 = [1; 0; 0];
   if L(n) == 0
      V2 = [0; 0;1];
   end
   if L(n) ~= 0
       V2 = [0; 1/K(n);0 ] - [0 ;0; 1/L(n)];
   end
   normalV(:,n) = cross(inv(B)*V1,inv(B)*V2);
end

normalV(:,n) = normalV(:,n)/norm(normalV(:,n));

end


h = normalV(1,:);
k = normalV(2,:);
l = normalV(3,:);

disp('In the orthogonal xyz coordinate system the plane normals are:')
disp([h; k; l]);

B = [ 1 0 0 ; 0 1 0 ; 0 0 1 ]; %MATRIX OF ORTHOGONALISATION;
%ALREADY DONE, SO B = THE IDENTITY MATRIX FROM NOW ON

end



%POLE FIGURE CALCULATIONS CONTINUED: EITHER LATTICE:
if CALC ~= 0 & CALC ~= -1

sAv=size(A);
sA=sAv(1,2); %sA = NUMBER OF DATA POINTS
fejlpkt=[];
for n=1:sA %IDENTIFIES BAD POINTS
   fejl=A(4,n)-A(5,n);
   if fejl > 4
      fejlpkt=[fejlpkt; n];
   end
end
antfejl=size(fejlpkt);
for n=1:antfejl(1) %THROWS BAD POINTS AWAY
   A(:,fejlpkt(n))=[];
   fejlpkt=fejlpkt-1;
end
A([4 5], :)=[]; %4. AND 5. ROW NO LONGER NECESSARY

%CONSTRUCTING g MATRIX FROM EULER ANGLES FOR EACH DATA POINT:
sA=size(A);
sA=sA(1,2);
for n = 1:sA;
   phi1 = A(1,n);
   phi = A(2,n);
   phi2 = A(3,n);
   phi1 = phi1 / 360 * (2 * pi);
```

```
   phi = phi / 360 * (2 * pi);
   phi2 = phi2 / 360 * (2 * pi);
   g(:, 1, n) = [ (cos(phi1)*cos(phi2)-sin(phi1)*sin(phi2)*cos(phi)) (-cos(phi1)*sin(phi2)-
sin(phi1)*cos(phi2)*cos(phi)) (sin(phi1)*sin(phi)) ];
   g(:, 2, n) = [ (sin(phi1)*cos(phi2)+cos(phi1)*sin(phi2)*cos(phi)) (-
sin(phi1)*sin(phi2)+cos(phi1)*cos(phi2)*cos(phi)) (-cos(phi1)*sin(phi)) ];
   g(:, 3, n) = [ (sin(phi2)*sin(phi)) (cos(phi2)*sin(phi)) (cos(phi)) ];
end

%POLES ARE PLANE NORMALS THROUGH (0,0,0).
%POLE VECTORS TO A POINT ON THE UNIT SPHERE ARE v = U ROT B (h,k,l) / |v|.
%HERE B ORTHOGONALISES LATTICE VECTORS (h,k,l) (WHICH ARE IN CUBIC OR
%HEXAGONAL BASIS) AND U IS AN (ORTHONORMAL) ROTATIONAL MATRIX, U = g',
%CALCULATED FROM THE EULER ANGLES. U IS THE ROTATION OF THE GRAIN FROM
%(RD,TD,ND).
%ROT IS THE ROTATION OF THE xy-PLANE FROM RD-TD.

ROT = [cos(THETA*pi/180) -sin(THETA*pi/180) 0; sin(THETA*pi/180) cos(THETA*pi/180) 0; 0 0 1];

for retn=1:sSYMM
  recvek = B*[h(retn); k(retn); l(retn)];
  for n = 1:sA
    G=g(:,:,n);
    U=G';
    V=inv(ROT)*U*ROT*recvek;
    %VECTORS POINTING TO THE LOWER HALF OF THE UNIT SPHERE ARE INVERTED
    %TO POINT TO THE UPPER HALF OF THE SPHERE:
    if V(3)<0
      V(1)=-V(1);
      V(2)=-V(2);
      V(3)=-V(3);
    end
    v(:,n)=V;
  end
  %NOW WE USE THE STEREOGRAPHIC PROJECTION ON THE NORMALISED VECTORS ON
  %THE UPPER UNIT SPHERE, GETTING FROM EACH OF THEM A POINT ON THE UNIT
  %CIRCLE IN THE xy-PLANE. pole IS A MATRIX CONTAINING THE NORMALISED
  %VECTORS AS COLUMNS, AND w IS A MATRIX CONTAINING THE PROJECTED
  %(NORMALISED) VECTORS AS COLUMNS.
  m=sA*(retn-1);
  for n = 1:sA
    V = v(:,n);
    V = V / norm(V); %NORMALISATION OF V
    pole(1,m+n) = V(1);
    pole(2,m+n) = V(2);
    pole(3,m+n) = V(3);
    w(1,m+n)=V(1)/(1+V(3));
    w(2,m+n)=V(2)/(1+V(3));
    w(3,m+n)=0;
  end
end
end



%CALCULATIONS FOR DENSITY PLOTS:
if (DENSITY_CONTOUR_PLOT == 1 | DENSITY_COLOR_PLOT == 1) & CALC ~= -1

N=20; %NUMBER OF CALCULATIONS ~ N^2; N=20 IS PROBABLY FINE.
angvar = SMOOTH*(pi/180); %ANGULAR VARIATION ON THE MEASURED POINTS.
%angvar MUST BE LESS THAN APPROXIMATELY 0.04 TO GET wconc (BELOW) LARGER
%THAN 5, WHICH IS NECESSARY FOR THE IMPLIED APPROXIMATION TO THE WATSON
%DISTRIBUTION TO BE VALID.

%NOW TO CALCULATE THE DENSITIES density(a,b) FOR POINTS (a,b) IN THE POLE
%PLANE BY CALCULATING THE CORRESPONDING DENSITIES DENS(x,y,z) ON THE POLE
%SPHERE (IN POINTS GIVEN BY THE INVERSE STEREOGRAPHIC PROJECTION). THE
%DENSITIES ON THE POLE SPHERE ARE GIVEN BY AN APPROXIMATION TO THE WATSON
%DISTRIBUTION:
%DENS(x,y,z) = K(wconc) * SUM (exp (wconc * ( (x,y,z) * (x0,y0,z0) )^2))
%WHERE (x,y,z) IS A POINT ON THE UNIT SPHERE, (x0,y0,z0) IS A POLE, wconc
%IS THE WATSON CONCENTRATION (DETERMINES THE SMOOTHING-OUT), AND K(wconc)
%IS A CONSTANT. THE SUM IS TAKEN FOR ALL POLES (x0,y0,z0).
```

```
%FOR EACH POINT (a,b) WE CALCULATE TEMPORARY VARIABLES:
%DENS = K * POLSUM
%BUT K IS ALWAYS THE SAME AND IS CALCULATED ONLY ONCE.

%CALCULATIONS FOR wconc AND mindot:
sp=size(pole);
wconc = (1-pi/4)/angvar;
if wconc>700
  %TOO LARGE wconc => mindot ALMOST 1 AND NEARLY ALL CONTRIBUTIONS
  %TO THE SUMMATION BELOW ARE DROPPED; WE DON'T WANT THAT!
  wconc=700;
end
if wconc < -log(eps)
  wconc=-log(eps); %TO AVOID mindot = sqrt(NEGATIVE NUMBER)
end
mindot = sqrt((1 + log(eps)/wconc)); %BETWEEN 0 AND 1; IN THE SUMMATION
%BELOW TERMS ARE NEGLECTED WHEN THE DOT-PRODUCTS IN THE EXPONENTIALS ARE
%LESS THAN mindot

%CALCULATION OF K(wconc) - STRICTLY SPEAKING UNNECESSARY DUE TO LATER
%NORMALISATION (K MAY BE SET TO 1), BUT HERE IT IS:
r = 0:0.0001:1;
nmax = 10000;
int_fkt = 0;
for n=1:nmax
  fkt(n) = 4 * pi * exp(wconc * r(n).^2);
  int_fkt = int_fkt + fkt(n);
end
int_fkt=int_fkt/nmax;
K = 1/int_fkt;

%WE NOW RUN THROUGH POINTS (a,b) IN THE POLE PLANE, CALCULATES TEMPORARY
%VARIABLES POLSUM AND DENS (ON THE POLE SPHERE), AND CALCULATES
%density(a,b) IN THE POLE PLANE:
for a=-N:N-1
  for b=-N:N-1
    if (a/N).^2 + (b/N).^2 <=1
      %PROJECTING (a,b) TO THE POLE SPHERE:
      z = (1-((a/N).^2 + (b/N).^2))/(1+((a/N).^2+(b/N).^2)); %NOTE z > 0
      x = (1+z)*(a/N);
      y = (1+z)*(b/N);
      %CALCULATES POLSUM - MANY POLES CONTRIBUTE NEGLIGIBLY AND MAY BE
      %IGNORED:
      POLSUM=0;
      for n=1:sp(2)
        dot = [x y z] * [pole(1,n); pole(2,n); pole(3,n)];
        if abs(dot) >= mindot
          POLSUM = POLSUM + exp(wconc * dot.^2);
       else
          POLSUM = POLSUM + 1;
        end
      end
      DENS = K * POLSUM;
    else
      DENS = 0;
    end
    density(a+N+1,b+N+1) = DENS;
  end
end
density = density / max(max(density));

%CALCULATES THE DENSITY PLOT INTERPOLATIONS:
X=-1:2/(2*N-1):1;
Y=-1:2/(2*N-1):1;
[xi,yi]=meshgrid(-1:0.001:1);
zi=interp2(X,Y,density,xi,yi,'*bicubic');
for n=-1000:1000
  for m=-1000:1000
    cst = (n/1000).^2+(m/1000).^2;
    if cst > 1
      zi(n+1001, m+1001) = NaN;
    end
  end
end
```

```
end


%DRAWING POINT, COUNTOUR AND COLOR PLOTS:
for n=1:3
  PLOT = 0;
  switch n
  case 1
    if POINT_PLOT == 1
      figure(1);
      axis([-1 1 -1 1])
      sw = size(w);
      sw = sw(2);
      for n=1:sw
        W1(n)=w(1,n);
        W2(n)=w(2,n);
        hold on
        plot(W1(n), W2(n))
      end
      PLOT = 1;
    end
  case 2
    if DENSITY_CONTOUR_PLOT == 1
      figure(2);
      contour(yi,xi,zi)
      colorbar
      PLOT = 1;
    end
  case 3
    if DENSITY_COLOR_PLOT == 1
      figure(3);
      pcolor(yi,xi,zi)
      shading interp
      colorbar
      PLOT = 1;
    end
  end
  if PLOT == 1 %THIS PART FOR ALL PLOT TYPES
    XX=-1:0.001:1;
    YY=sqrt(1-XX.^2);
    hold on
    plot(XX, YY, '-')
    hold on
    plot(XX, -YY, '-')
    axis equal;
    xlabel('TD')
    ylabel('RD')
    if LATTICETYPE == 1
      hkl = [];
      sh0 = size(h0);
      sh0 = sh0(2);
      for n=1:sh0
        h0n = int2str(h0(n));
        k0n = int2str(k0(n));
        l0n = int2str(l0(n));
        hkl = [hkl, ' (', h0n, ' ',k0n, ' ',l0n, ') ' ];
      end
      title([FILENAME,EXTENSION,'   ',date,'   bcc/ fcc   ', hkl])
    end
    if LATTICETYPE == 2
      hkil = [];
      sh1 = size(h1);
      sh1 = sh1(2);
      for n=1:sh1
        h1n = int2str(h1(n));
        k1n = int2str(k1(n));
        i1n = int2str(i1(n));
        l1n = int2str(l1(n));
        C = num2str(C);
        hkil = [hkil ' (' h1n ' ' k1n ' ' i1n ' ' l1n ') ' ];
      end
```

```
      title([FILENAME,EXTENSION,'  ',date,'  hcp  ', hkil,'  C=',C])
    end
  end
end
```

Title and authors

Texture in titanium alloys (in Danish)

Gregers Alexander Kaat

ISBN	ISSN
87-550-3107-2; 87-550-3108-0 (Internet)	0106-2840

Department or group	Date
Materials Research Department	August 2002

Groups own reg. number(s)	Project/contract No(s)

Sponsorship

Pages	Tables	Illustrations	References
23	0	4	6

Abstract (max. 2000 characters)

Existing software for processing of EBSD-data has been further developed with the aim of being able to handle hcp materials. The specific problem has been texture in thin titanium sheets, which, based on x-ray investigations, were expected to have a weak texture.

The newly developed program has been able to handle measured EBSD-data and to plot these as polefigures . New EBSD-measurements have been carried out on titanium alloys supplied by Alfa Laval, and the results show a strong texture in the alloys.

Descriptors INIS/EDB